

**Design and Analysis of Algorithms**

**Project Report**

**Submitted to:**

Dr. Nasir Uddin

**GROUP MEMBERS:**

Anas Saleem [22K-0500]

Ashar Zamir [22K-4241]

Bilal Ahmed Khan [22K-4525]

**Abstract:**

This report details the implementation and performance analysis of two divide-and-conquer algorithms: Closest Pair of Points and Integer Multiplication. The aim is to assess their efficiency and correctness across a range of datasets. Various input datasets, generated with different complexities, were processed by the algorithms. The results, focusing on accuracy, time efficiency, and scalability, were recorded. The findings align with the expected performance trends, providing valuable insights into how each algorithm behaves under diverse conditions.

**Example:**

This project implements and evaluates divide and conquer algorithms to solve the Closest Pair of Points and Integer Multiplication problems. Using random input datasets, the solutions were tested for accuracy and efficiency. The results demonstrate the effectiveness of the divide-and-conquer approach in handling large datasets efficiently.

**Introduction:**

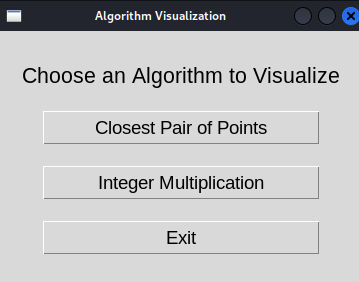
Divide-and-conquer is a fundamental algorithm design paradigm that solves problems by dividing them into smaller subproblems, solving these recursively, and then combining their solutions. This report focuses on two such algorithms:

1. **Closest Pair of Points**: This algorithm identifies the pair of points with the smallest Euclidean distance in a given dataset. Its time complexity is O(nlogn), making it efficient for large datasets. Its recurrence relation is T(n)=2T(n/2) + O(nlogn).
2. **Integer Multiplication**: This algorithm, based on Integer's method, efficiently multiplies large integers using divide and conquer principles. Its time complexity is O(n^log2(3)). Its recurrence relation is T(n)=3T(n/2)+O(n).

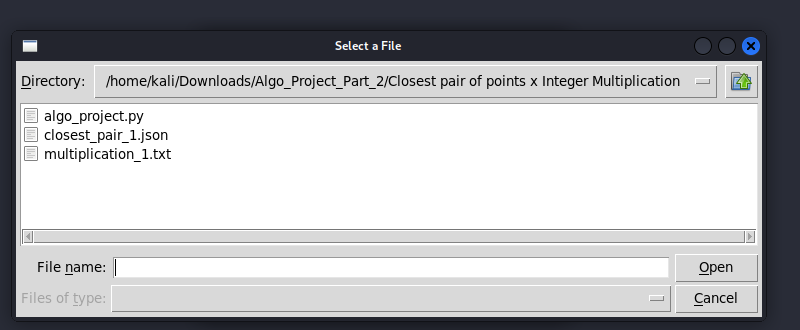
The aim is to implement these algorithms, test them on datasets with input sizes exceeding 100, and analyze their correctness and efficiency. The datasets were randomly generated to simulate diverse complexities, ensuring a robust evaluation.

Through this report, the application of divide-and-conquer is demonstrated in computational geometry and arithmetic, emphasizing their practical relevance in solving real-world problems.

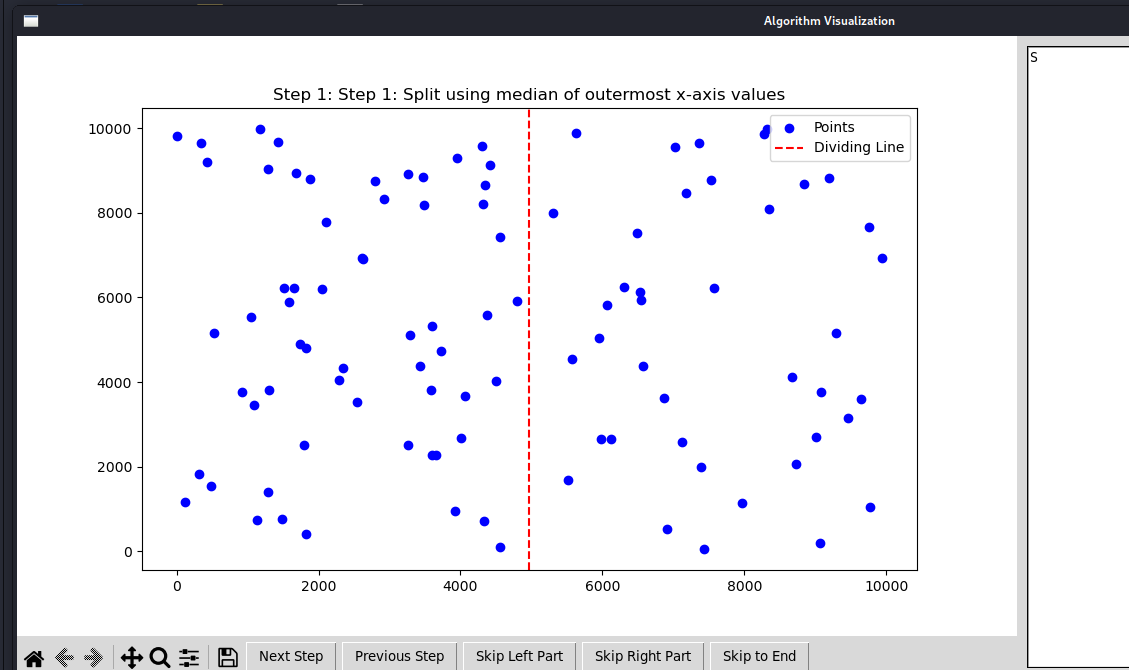
**YOUR PROPOSED SYSTEM:**



this dialog box will appear when we run the program.



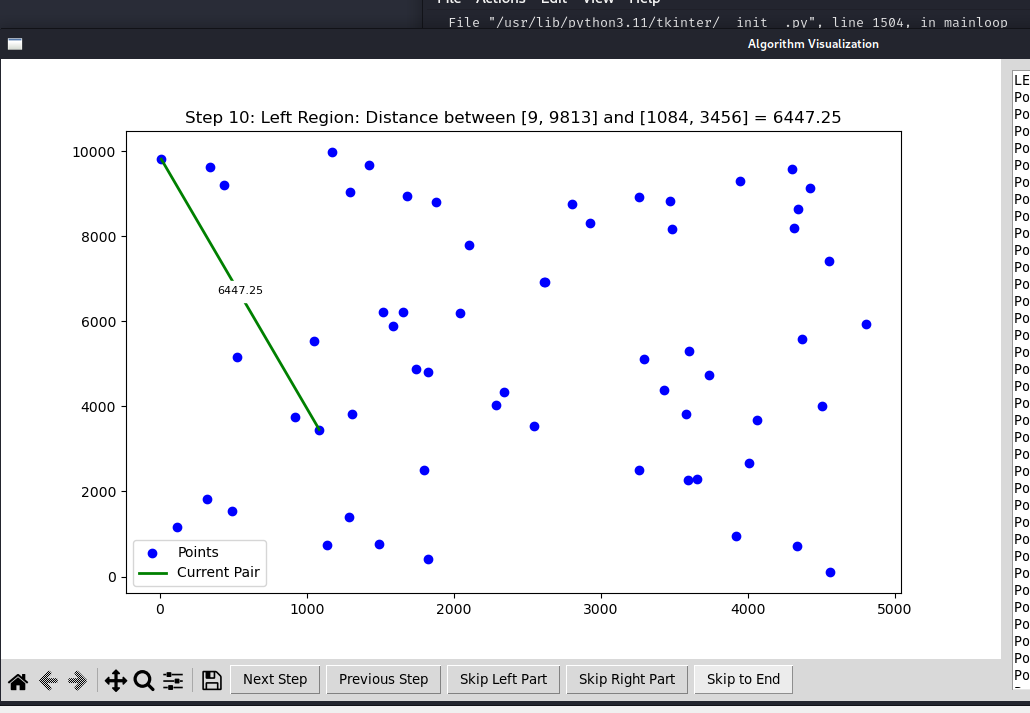
If we select Closest Pair of Points button then a box opens from which we have to choice the closest\_pair\_1.Json file

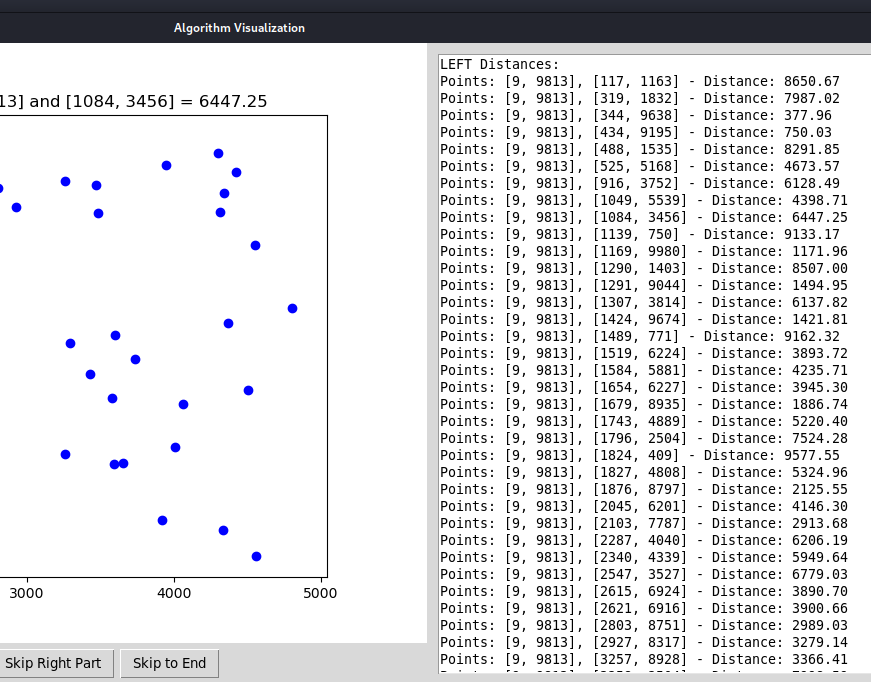


When we open the json file we will see the visual representation of the data.

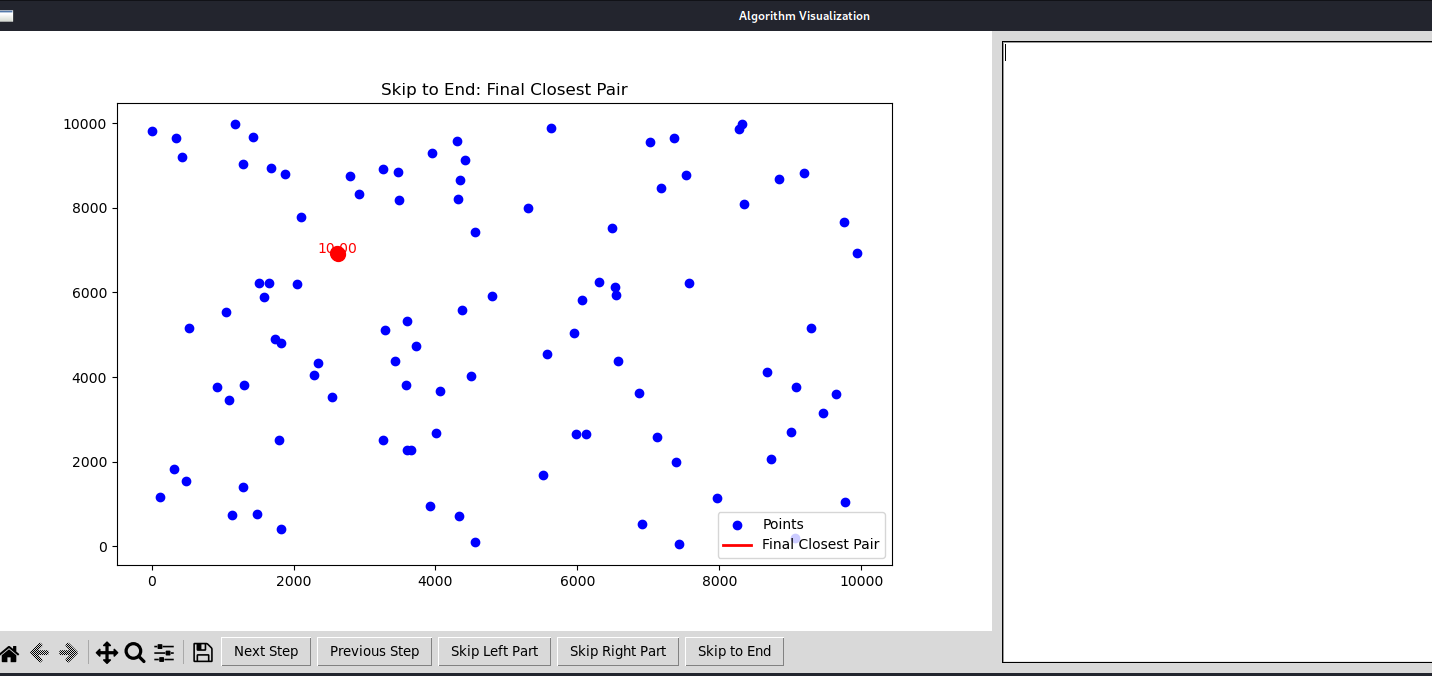


We can navigate through 100s of steps through these options at botton

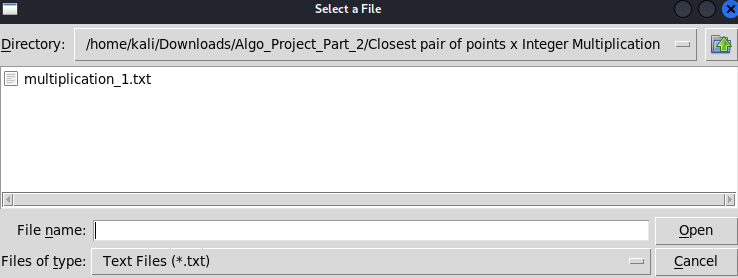




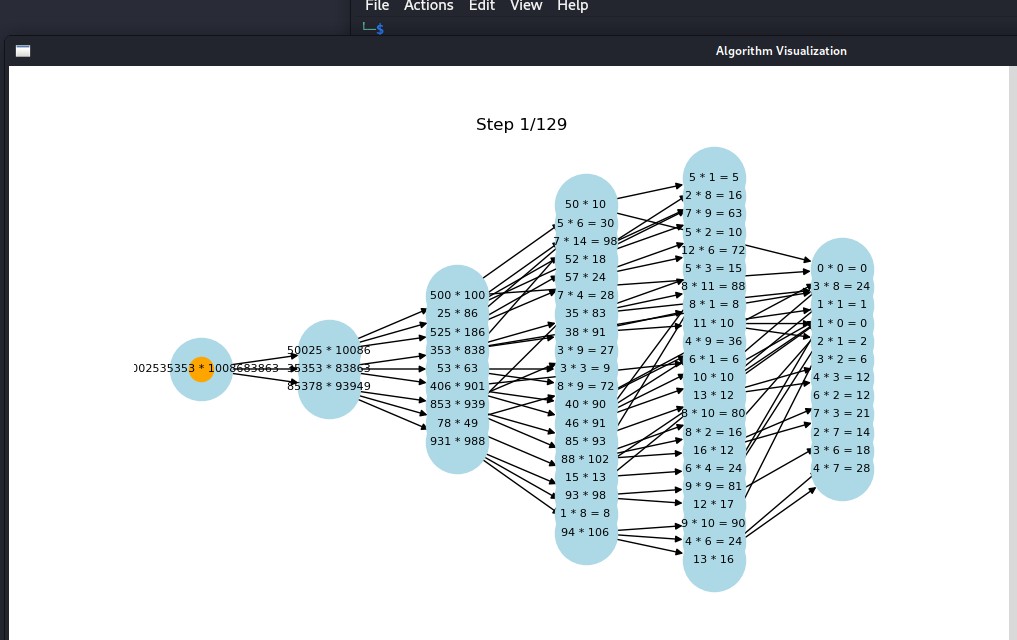
On the right Hand side we can see all the distances among which calculation is taking place.



Finally if we click on skip to end we can see the end outcome bypassing all the steps if not necessary.



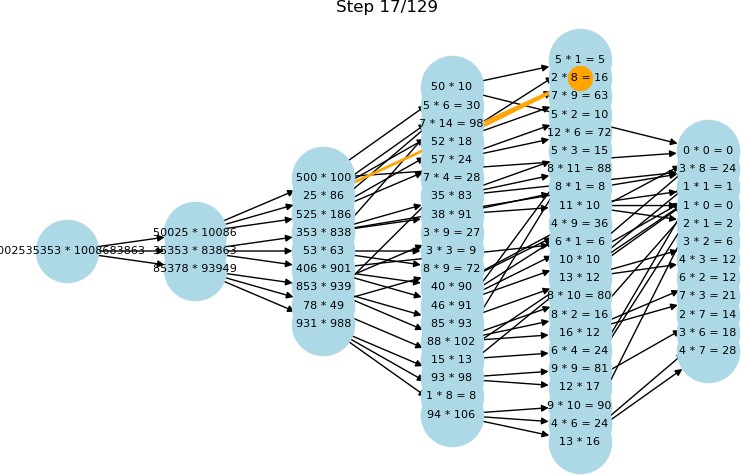
If we click on Integer Multiplcation we can see an icon\_box showing the dataset for it.

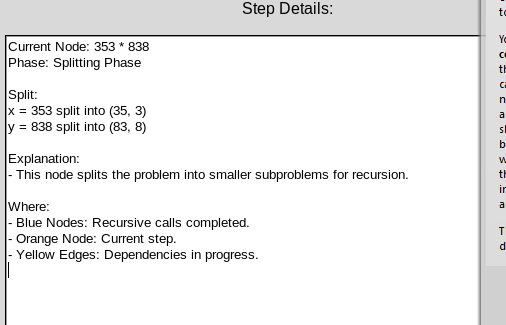


After clicking on the dataset we can see the Visualization with the required steps

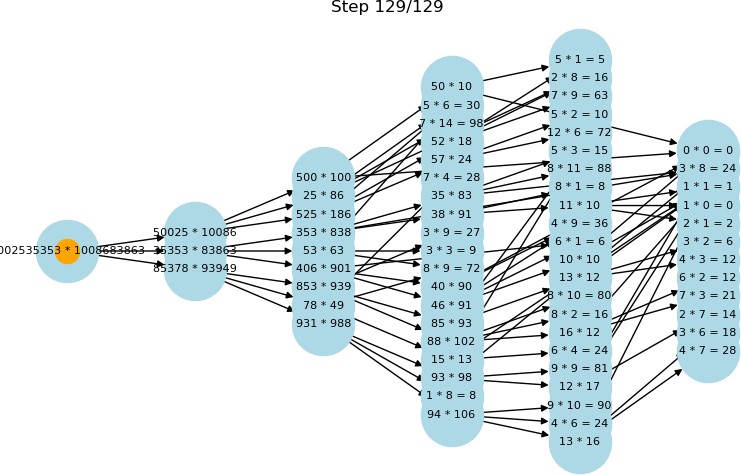


We can navigate to other steps and even restart the program





On the right side we can see the details explanation and node info.



**Experimental Setup:**

# Details Of Dataset:

|  |  |  |
| --- | --- | --- |
| **Point ID** | **X** | **Y** |
| 1 | 1824 | 409 |
| 2 | 4506 | 4012 |
| 3 | 3657 | 2286 |
| 4 | 1679 | 8935 |
| 5 | 1424 | 9674 |
| 6 | 6912 | 520 |
| 7 | 488 | 1535 |
| 8 | 3582 | 3811 |
| 9 | 8279 | 9863 |
| 10 | 434 | 9195 |
| 11 | 3257 | 8928 |
| 12 | 6873 | 3611 |
| 13 | 7359 | 9654 |
| 14 | 4557 | 106 |
| 15 | 2615 | 6924 |
| 16 | 5574 | 4552 |
| 17 | 2547 | 3527 |
| 18 | 5514 | 1674 |
| 19 | 1519 | 6224 |
| 20 | 1584 | 5881 |
| 21 | 5635 | 9891 |
| 22 | 4333 | 711 |
| 23 | 7527 | 8785 |
| 24 | 2045 | 6201 |
| 25 | 1291 | 9044 |

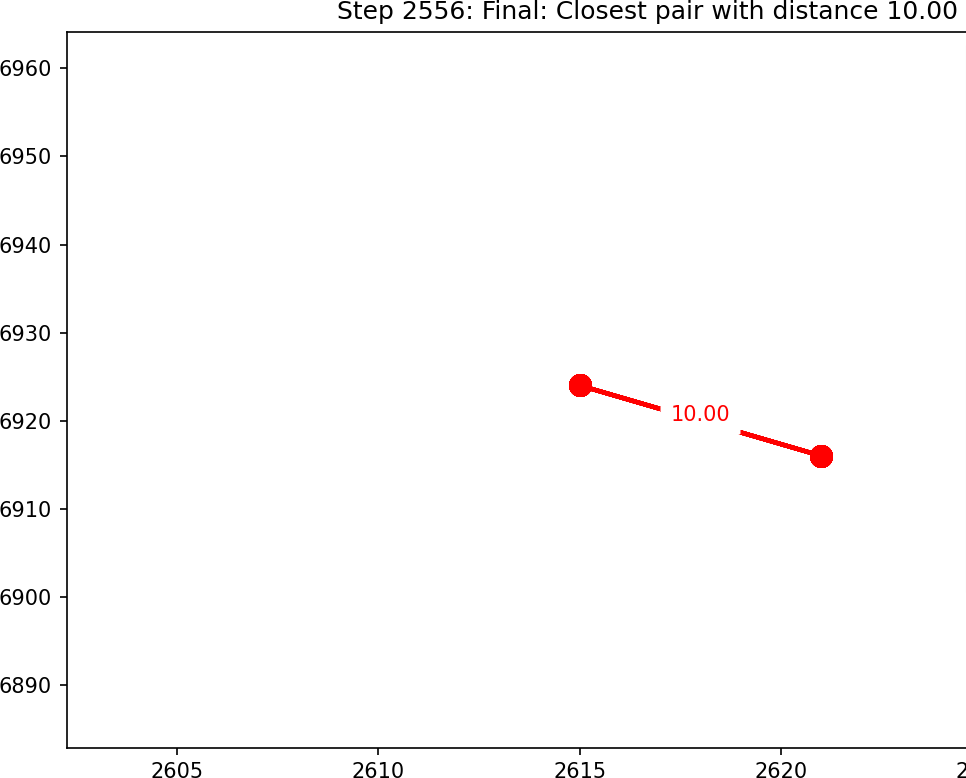
|  |  |  |
| --- | --- | --- |
| 26 | 4803 | 5925 |
| 27 | 9459 | 3150 |
| 28 | 1139 | 750 |
| 29 | 3733 | 4741 |
| 30 | 1307 | 3814 |
| 31 | 1654 | 6227 |
| 32 | 4554 | 7428 |
| 33 | 5977 | 2664 |
| 34 | 6065 | 5820 |
| 35 | 3432 | 4374 |
| 36 | 1169 | 9980 |
| 37 | 2803 | 8751 |
| 38 | 4010 | 2677 |
| 39 | 7573 | 6216 |
| 40 | 4422 | 9125 |
| 41 | 3598 | 5313 |
| 42 | 916 | 3752 |
| 43 | 525 | 5168 |
| 44 | 6572 | 4386 |
| 45 | 1084 | 3456 |
| 46 | 9292 | 5155 |
| 47 | 3483 | 8179 |
| 48 | 6482 | 7517 |
| 49 | 2340 | 4339 |
| 50 | 2287 | 4040 |
| 51 | 9197 | 8830 |
| 52 | 4304 | 9577 |
| 53 | 7019 | 9560 |
| 54 | 6543 | 5930 |

|  |  |  |
| --- | --- | --- |
| 55 | 3593 | 2266 |
| 56 | 8348 | 8085 |
| 57 | 1489 | 771 |
| 58 | 1796 | 2504 |
| 59 | 2621 | 6916 |
| 60 | 9771 | 1040 |
| 61 | 6304 | 6252 |
| 62 | 9763 | 7668 |
| 63 | 8669 | 4119 |
| 64 | 9064 | 188 |
| 65 | 1876 | 8797 |
| 66 | 4371 | 5573 |
| 67 | 1827 | 4808 |
| 68 | 7123 | 2591 |
| 69 | 7433 | 53 |
| 70 | 4315 | 8201 |
| 71 | 2927 | 8317 |
| 72 | 1743 | 4889 |
| 73 | 8317 | 9977 |
| 74 | 3258 | 2504 |
| 75 | 6126 | 2646 |
| 76 | 8837 | 8689 |
| 77 | 9 | 9813 |
| 78 | 5310 | 8005 |
| 79 | 319 | 1832 |
| 80 | 5947 | 5038 |
| 81 | 3923 | 949 |
| 82 | 3946 | 9295 |
| 83 | 1290 | 1403 |

|  |  |  |
| --- | --- | --- |
| 84 | 7962 | 1133 |
| 85 | 8727 | 2060 |
| 86 | 2103 | 7787 |
| 87 | 9007 | 2705 |
| 88 | 4342 | 8645 |
| 89 | 9938 | 6932 |
| 90 | 3470 | 8835 |
| 91 | 3295 | 5107 |
| 92 | 6537 | 6118 |
| 93 | 7177 | 8479 |
| 94 | 7397 | 1982 |
| 95 | 4061 | 3681 |
| 96 | 1049 | 5539 |
| 97 | 344 | 9638 |
| 98 | 9075 | 3770 |
| 99 | 9641 | 3608 |
| 100 | 117 | 1163 |

JSON files of these points are also given

# Results and Discussion: Closest Pairs Points:



**Integer Multiplication:**

|  |  |  |  |
| --- | --- | --- | --- |
| File\_name | Integer1 | Integer2 | Result |
| Mul1 | 6870440325  7124624392  0939857116  1990048319  8474116834  5629866975  3186841037  3494617344  1733372960  8595395221 | 82539811198496486  08496875166641711  24047075753237945  87448104641550974  67751115609134836  517556170094820 | 56708484733484335258733472595183506  49254708650927856740270984429853970  80243998788257819938824050553948148  51590917530551087336806161312115691  83358790232450403325437206595691003  8790834827704078944855220 |
| Mul2 | 5002535353 | 1008683863 | 5045976684658108000 |

# Discussion:

* Compare the observed results with the theoretical time complexities of the algorithms.

Example: "The Closest Pair algorithm's time increased quadratically as expected due to the O(nlogn) complexity."

* Discuss any interesting findings or trends.

Example: "For datasets with densely packed points, the Closest Pair algorithm performed significantly faster due to fewer recursive calls."

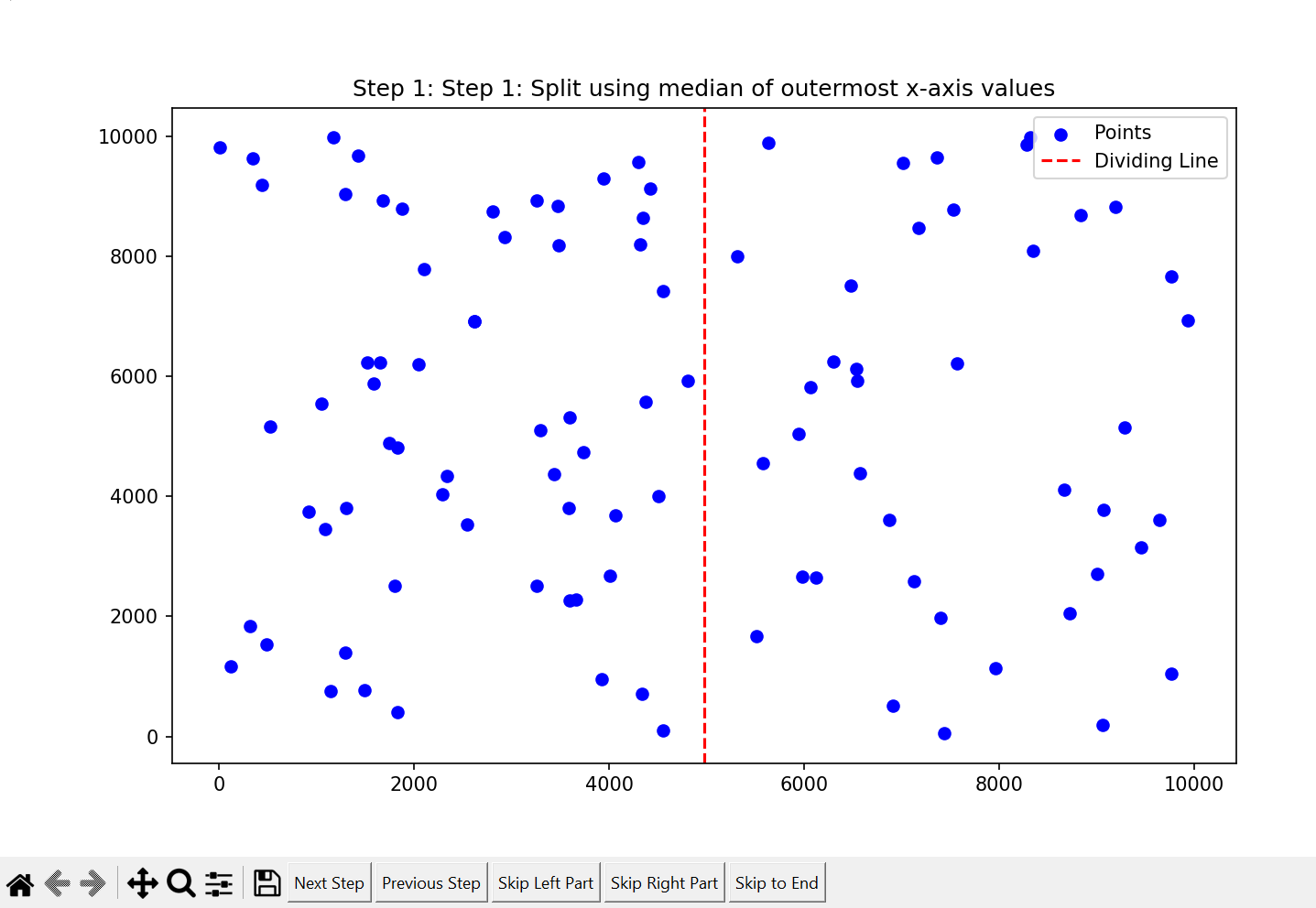
* Highlight potential improvements or limitations.

Example: "For extremely large inputs, the multiplication algorithm's efficiency was bottlenecked by the system's memory constraints."

**VISUALIZATION:**

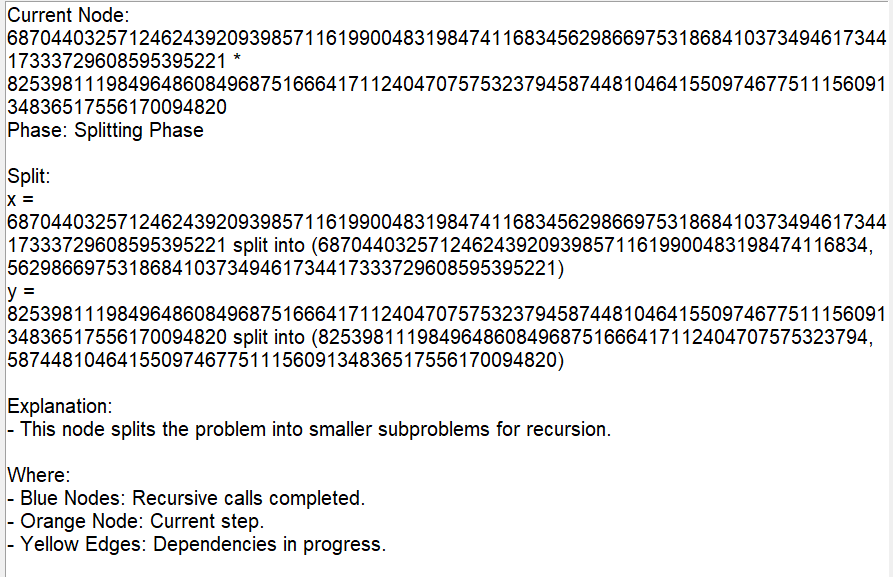
**Graph of Closest Pairs:**

**File = closest\_pair\_1.json**



**Steps of Integer Multiplication:**

**File = Multiplication\_1.txt**



**Conclusion:**

The divide-and-conquer paradigm proves its versatility and efficiency through the implementation of the Closest Pair of Points and Integer Multiplication algorithms. The results demonstrated that both algorithms produce accurate outputs while handling datasets of varying sizes and complexities. The Closest Pair of Points algorithm efficiently handled large datasets with minimal time complexity compared to brute force approaches, while the Integer Multiplication algorithm showcased its capability to manage high-precision arithmetic. These findings highlight the effectiveness of divide-and-conquer strategies in solving computationally intensive problems. Future work could explore parallelizing these algorithms to further enhance their performance.

**References:**

* Integer, A. (1962). Multiplication of Many-Digit Numbers by Means of Fast

*Algorithms.*

* Thomas H. Cormen et al, Introduction to Algorithms, 4th Edition
* Divide and Conquer Algorithm Design. GeeksforGeeks.
* Libraries or frameworks (e.g., Matplotlib).